Lab 12 Report

Jason Tolbert

The Pennsylvania State University

IST 894-001: Capstone Experience

Dr. Michael Bartolacci, Instructor

April 22nd, 2024

Table of Contents

General Overview *3*

Technical Overview *5*

References *7*

Screenshots *8*

General Overview

This lab introduces participants to several new methods of vulnerability analysis and security event monitoring.

The lab begins with an introduction to static code analysis — the practice of examining source code for bugs and other issues. In this case, participants are specifically focusing on using static code analysis to find vulnerabilities. Using MobSF, a tool for assessing the security of mobile applications (Kusreynada & Barkah, 2024), users perform static code analysis on an Android application designed specifically for the lab. After MobSF completes its analysis, users are able to view a report detailing the security risks found in the application's source code.

The lab then switches focus to the topic of web application vulnerabilities. By the time they participate in this lab, participants will have already a good deal of familiarity with this topic; however, this lab approaches it from an angle slightly different from what they are used to. Instead of exploiting a deliberately insecure app and observing the consequences of common vulnerabilities, they instead see an example of an app that successfully defends against one particular common vulnerability — arbitrary code execution. Participants make several attempts to perform arbitrary code execution in the app, only to discover that the app appropriately sanitizes user input, rendering their attacks futile.

Participants are able to see firsthand the benefits of proper input validation and secure coding practices (Wang et al., 2024).

The remainder of the lab focuses on incident detection and analysis. Participants investigate simulated security incidents using Splunk, a popular tool for searching and analyzing logs, to examine system logs for suspicious behavior. Participants also learn to detect phishing attacks in an exercise where they examine the headers and metadata of several emails to determine whether or not they originated from the source they claim to originate from. These exercises reinforce what participants should, by this point, already be familiar with: the importance of centralized monitoring in responding to attempted cyberattacks (Eze & Shamir, 2024; Messina et al., 2015).

Technical Overview

This lab introduces participants to several new methods of vulnerability analysis and security event monitoring.

The lab begins by introducing participants to static code analysis. Participants perform static code analysis on a deliberately-vulnerable APK using a containerized deployment of MobSF. After installing Docker and getting the MobSF container running, they upload the APK to MobSF and examine a report detailed the APKs permissions, exported services and activities, and behaviors it exhibits that have the potential to be dangerous.

The lab then turns its focus to web application vulnerabilities. Participants attempt to perform arbitrary code execution on a web application — something they have done before in prior labs. This time, however, the application has been programmed to properly sanitize user input, and no ACE vulnerability is actually present. Participants make several attempts to obtain user information through ACE — using commands like *id* and *whoami* — to no avail. They then find out exactly why their attacks failed by examining the application's PHP source, from which they learn that a basic string replacement algorithm is used for input sanitization.

The remainder of the lab focuses on incident detection and analysis. Participants use a locally hosted Splunk instance to examine an Apache access log. They are given a surface

level introduction to Splunk's proprietary Search Processing Language and use SPL commands to analyze the access log and identify suspicious behavior. Participants also learn to identify assess the legitimacy of emails and identify potential phishing attacks by examining email headers and metadata.

References

Eze, C. S., & Shamir, L. (2024). *Analysis and prevention of AI-based phishing email attacks* (No. arXiv:2405.05435). arXiv. https://doi.org/10.48550/arXiv.2405.05435

Kusreynada, S. U., & Barkah, A. S. (2024). Android apps vulnerability detection with static and dynamic analysis approach using MOBSF. *Journal of Computer Science and Engineering (JCSE)*, *5*(1), 46–63. https://doi.org/10.36596/jcse.v5i1.789

Messina, A., Fontana, I., & Giacalone, G. (2015). *Log monitoring and analysis with rsyslog and Splunk*. Unpublished. https://doi.org/10.13140/RG.2.1.2153.5128

Wang, X., Zhai, J., & Yang, H. (2024). Detecting command injection attacks in web applications based on novel deep learning methods. *Scientific Reports*, *14*(1), 25487. https://doi.org/10.1038/s41598-024-74350-3

Screenshots

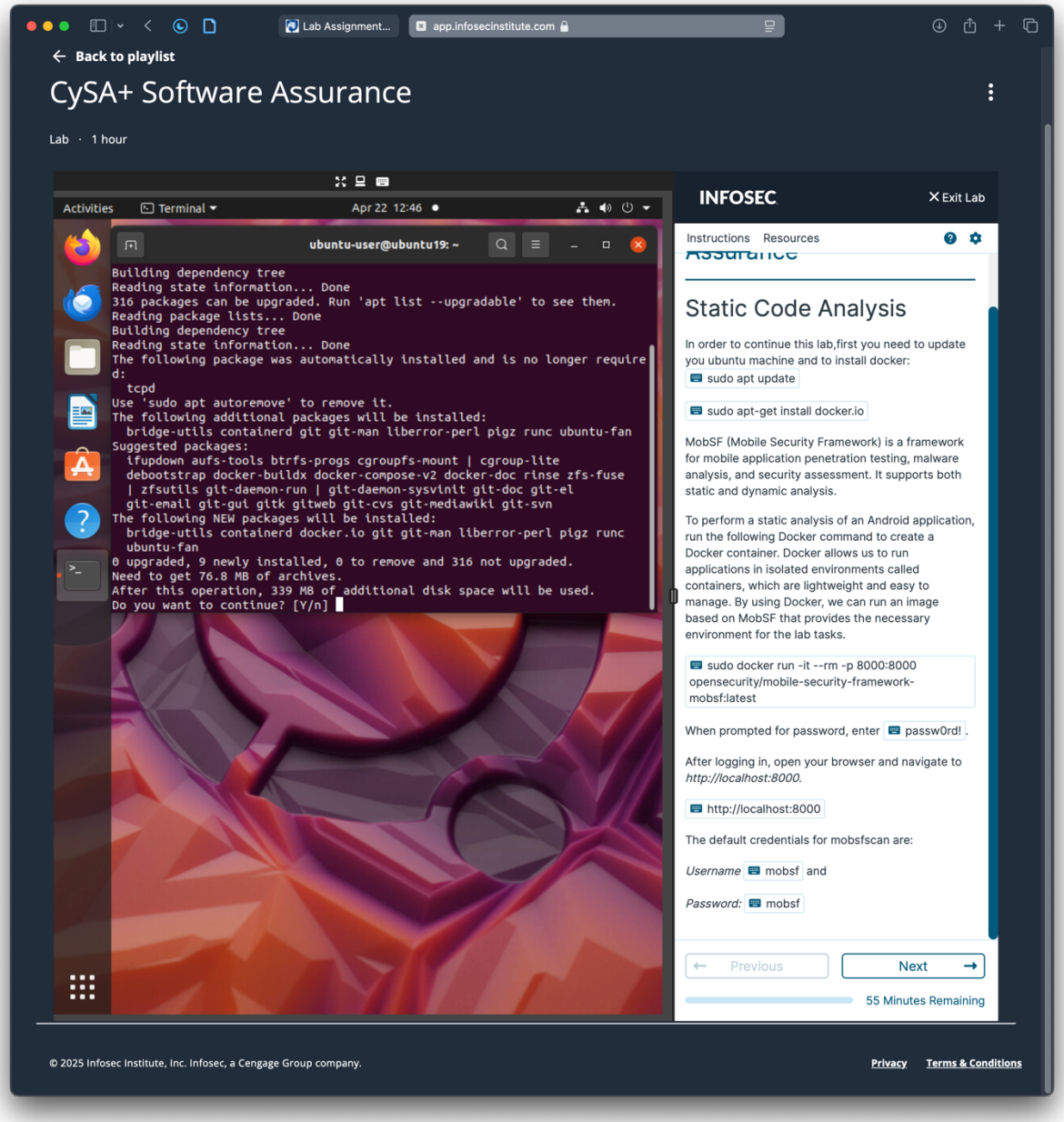

Figure 1. Installing Docker.

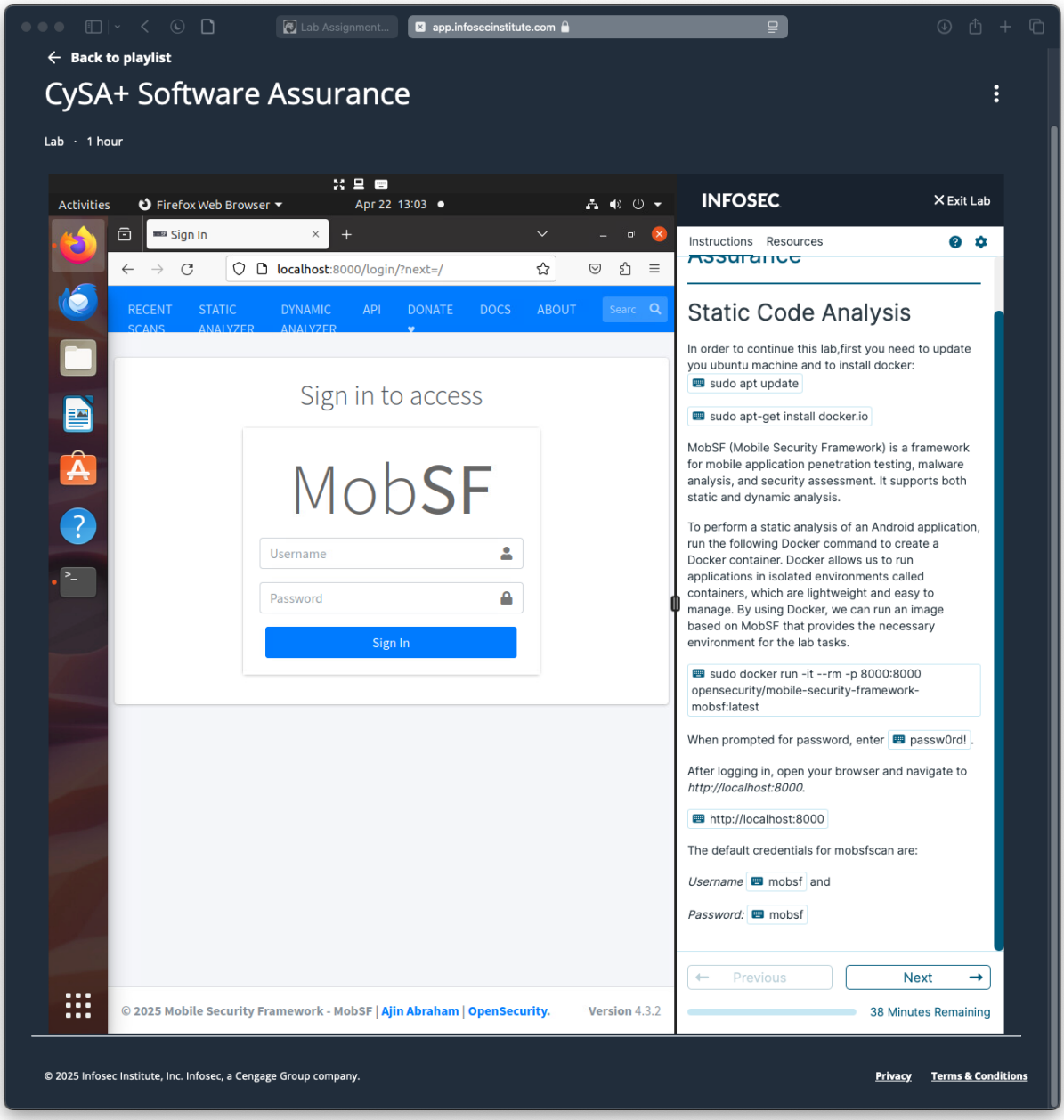

Figure 2. The MobSF login screen.

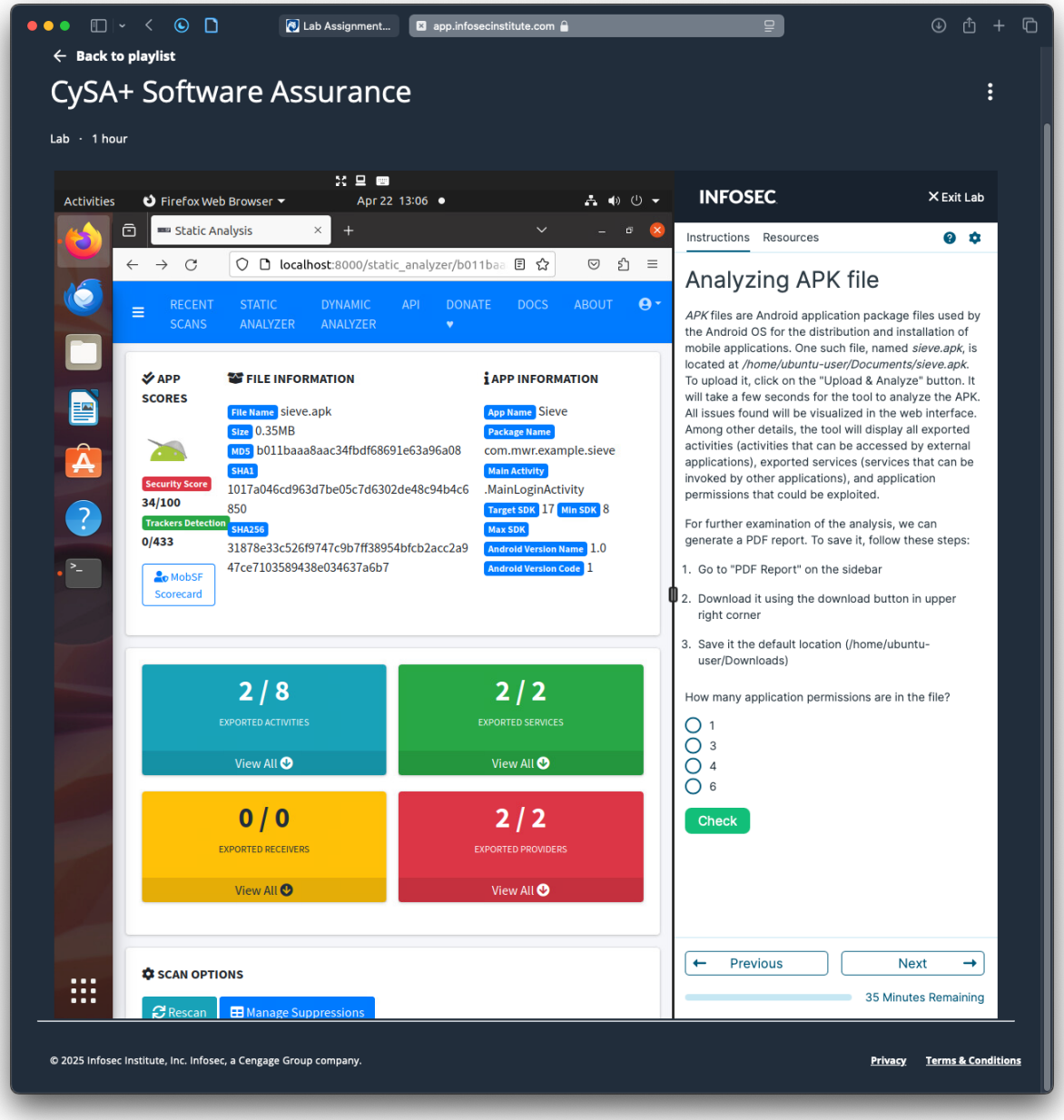

Figure 3. MobSF report for the file located at /home/ubuntu-user/Documents/sieve.apk.

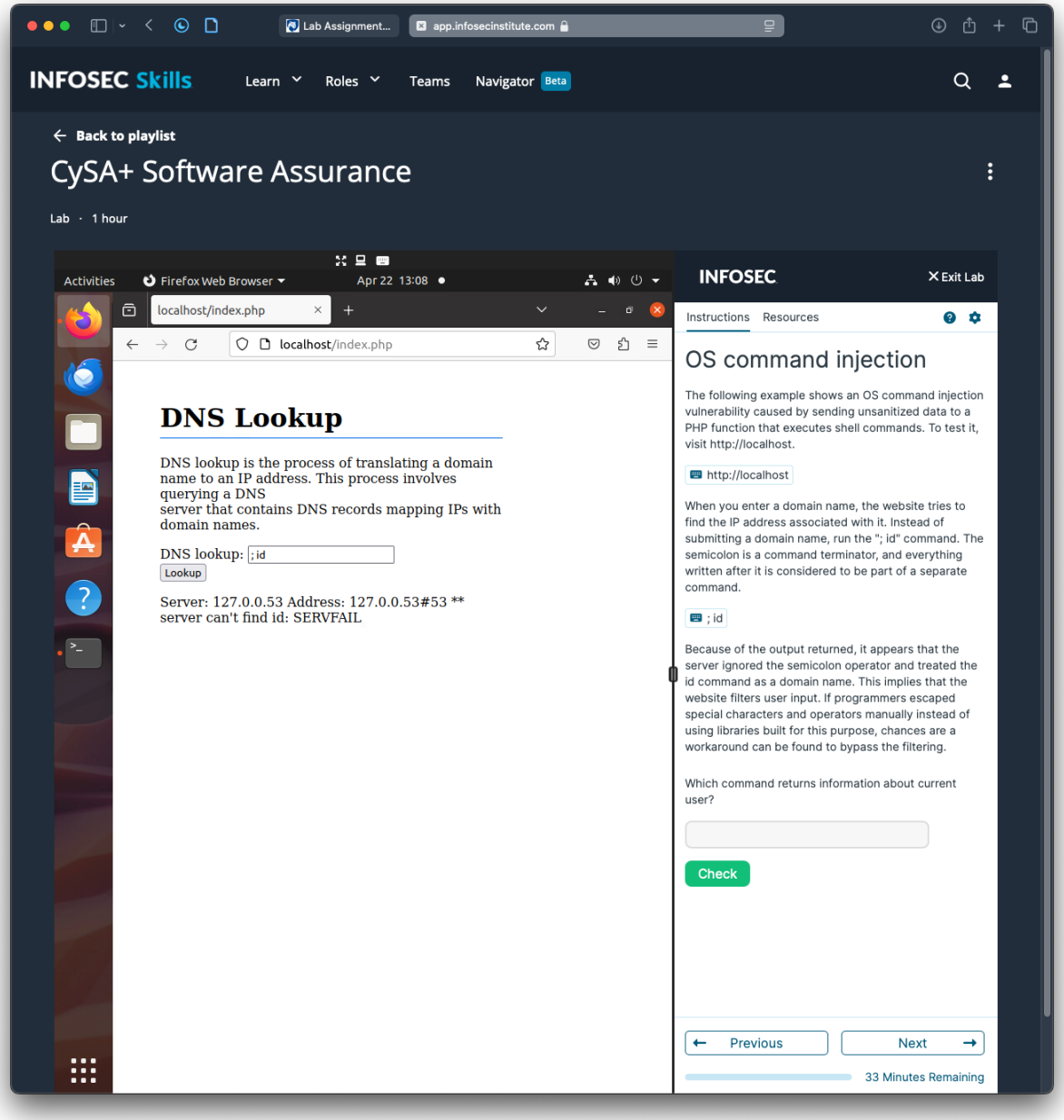

Figure 4. Demonstrating that the web application running on port 80 of the virtual machine sanitizes user input.

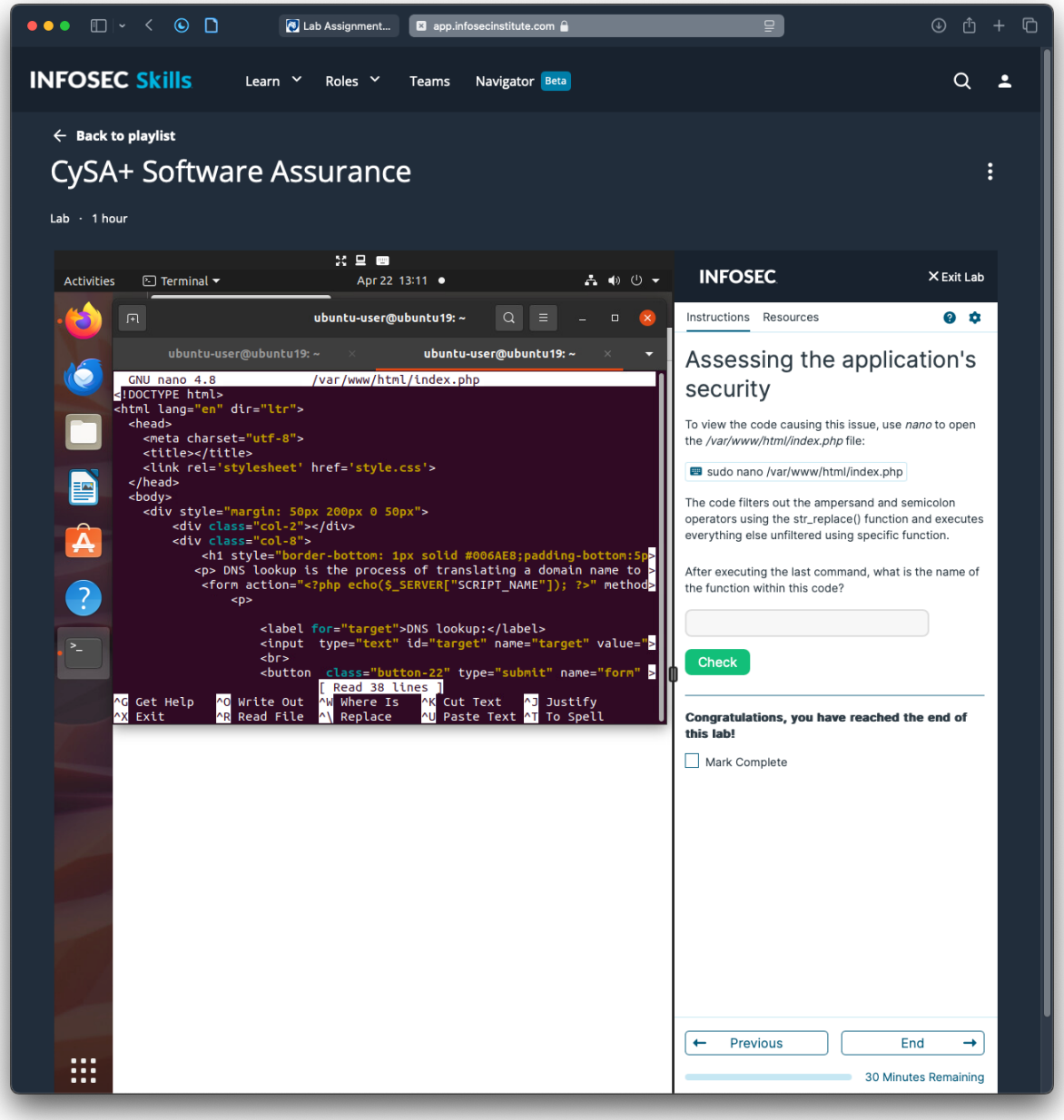

Figure 5. Examining the source code of the web application running on port 80 of the virtual machine.

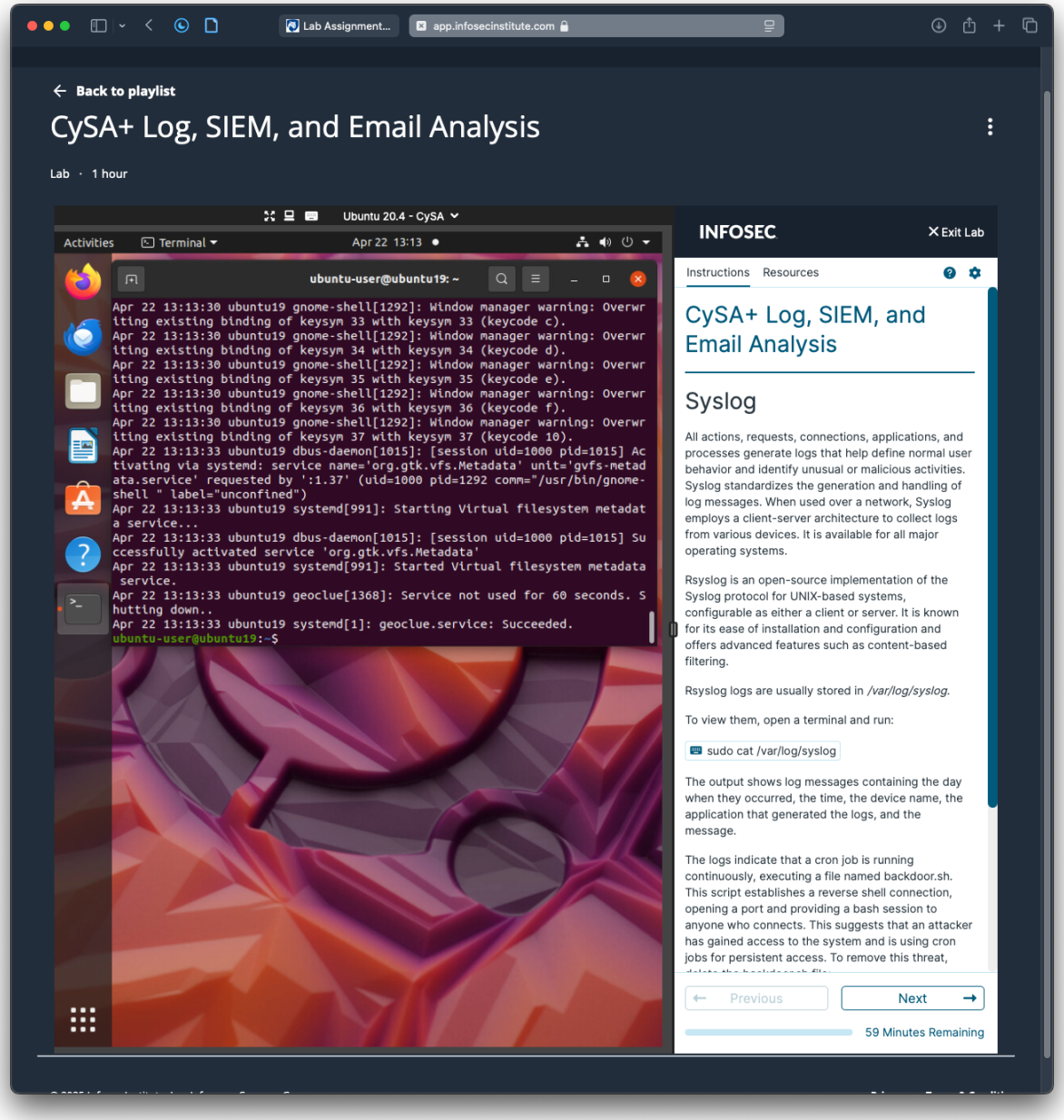

Figure 6. Viewing /var/log/syslog.

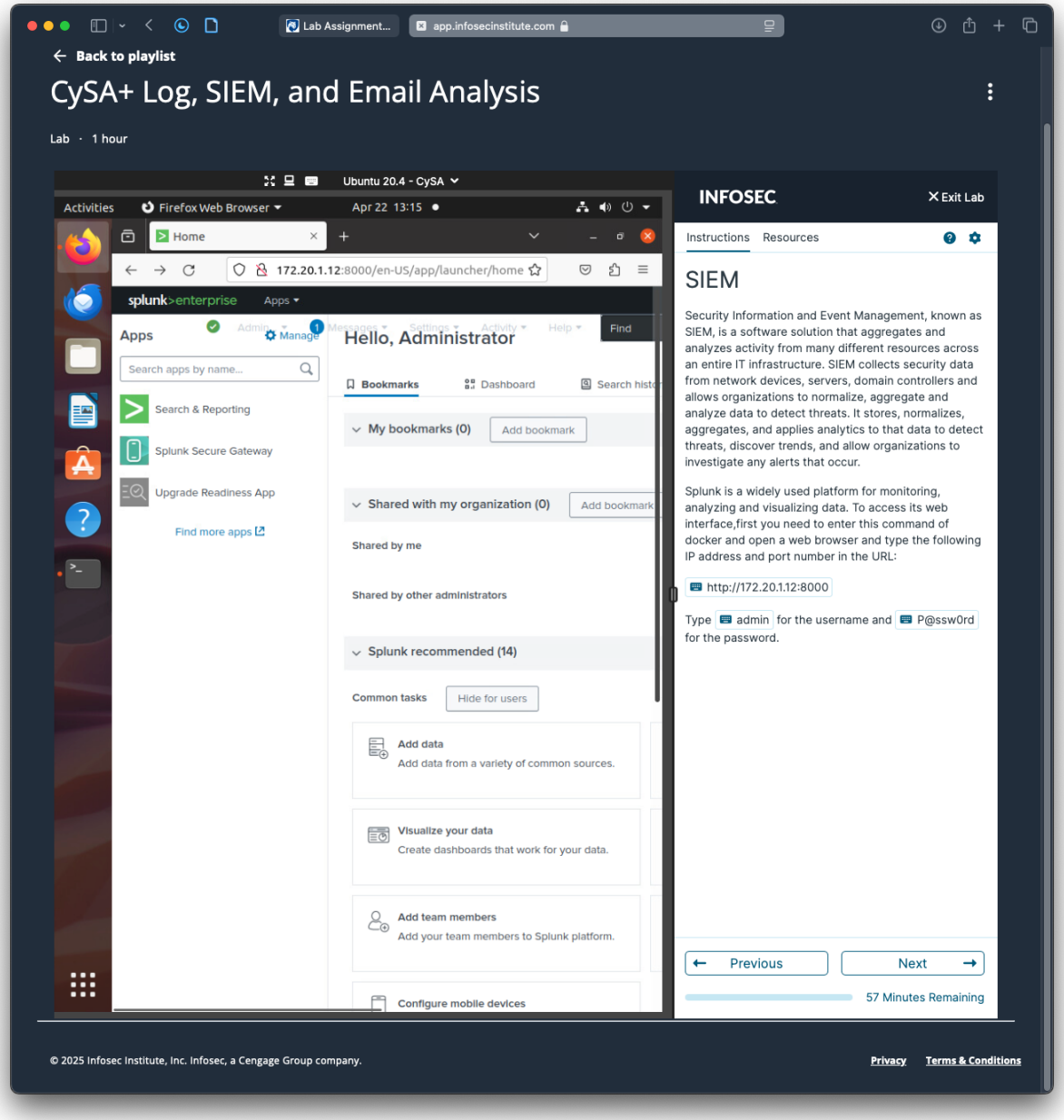

Figure 7. The Splunk user interface.

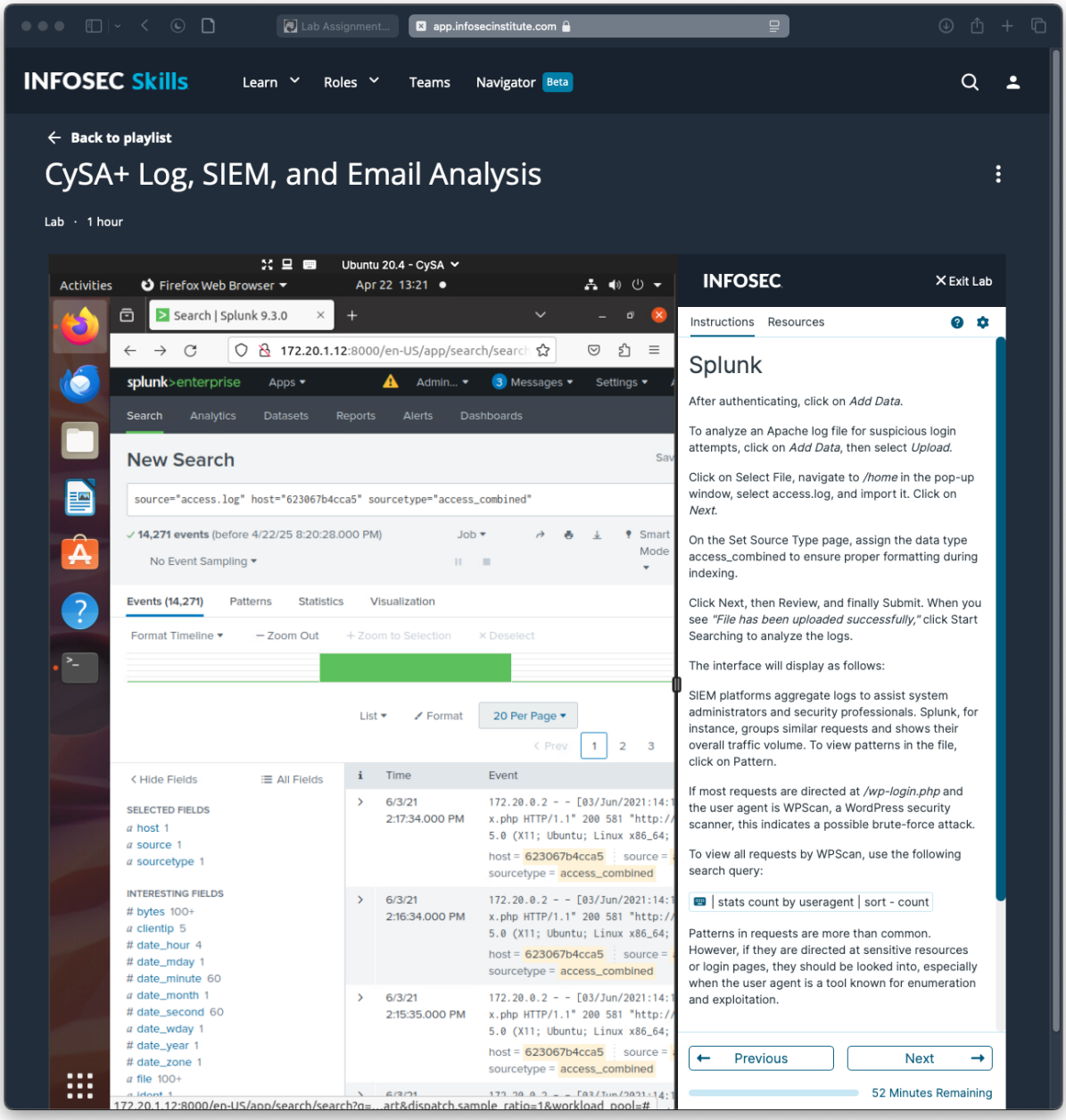

Figure 8. Searching /home/ubuntu-user/access.log in Splunk.

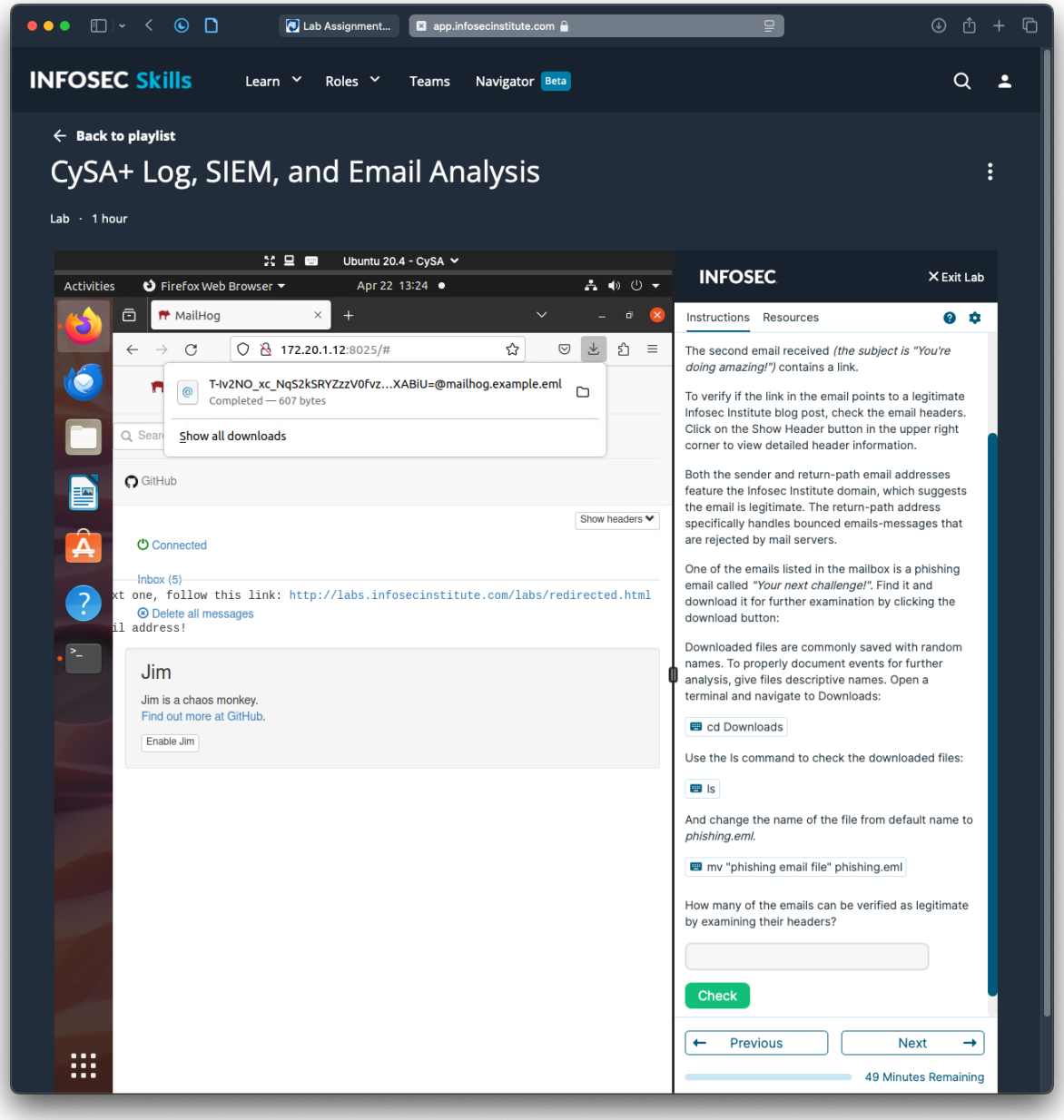

Figure 9. Downloading the email with the subject line "Your next challenge!".

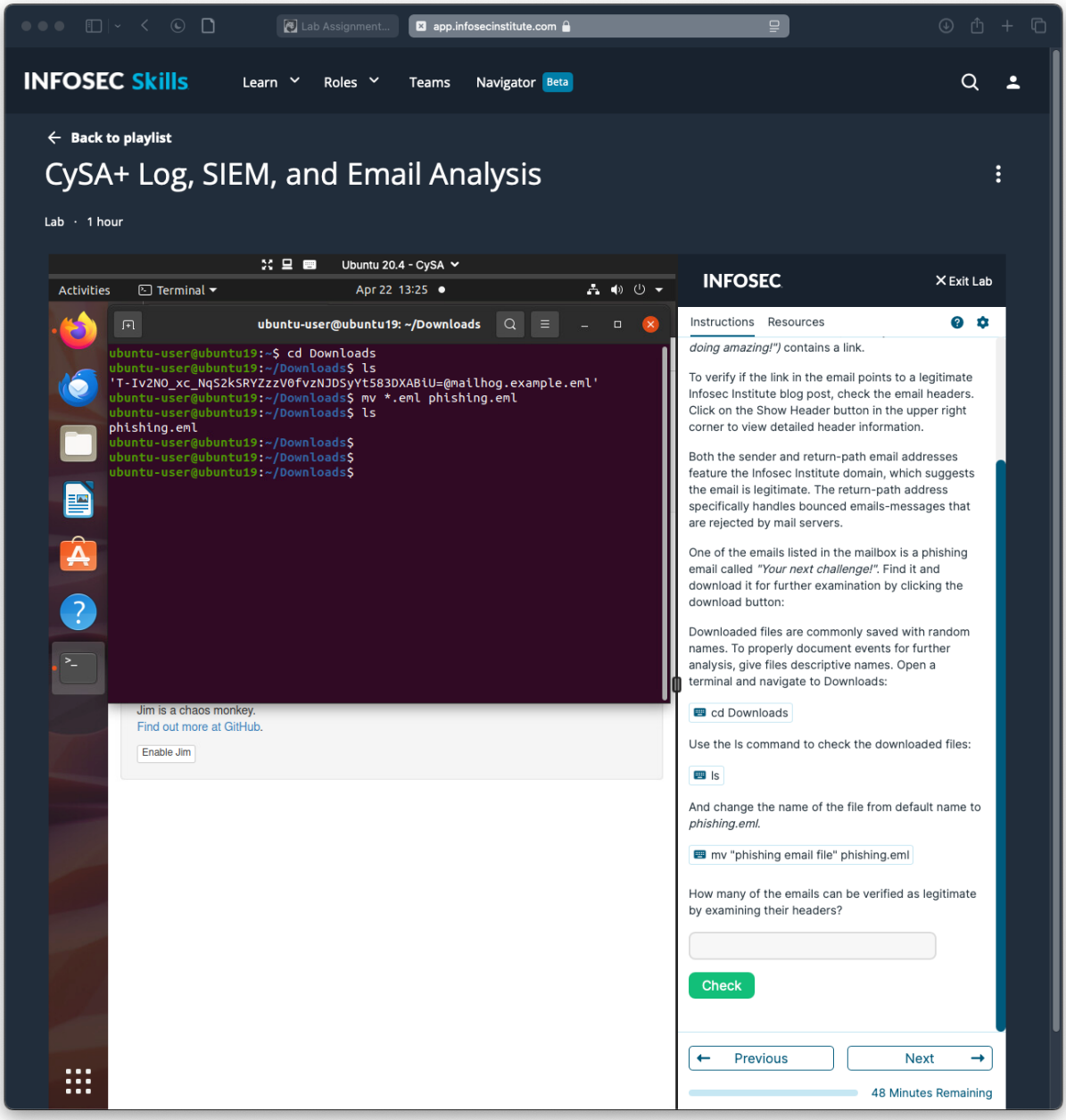

Figure 10. Renaming the downloaded email (see Figure 9) to "phishing.eml".